

Range Slider

Range Slider The Range Slider component provides a simple and elegant way to select a numeric value or range within a defined interval. It features smooth drag interactions, built-in accessibility, and easy integration with Angular forms for both template-driven and reactive use cases. The `<aava-slider>` component supports multiple modes such as single value , multi-range , and input-integrated sliders, making it adaptable to diverse UI needs. How to use ■ `import { AavaSliderComponent } from "@aava/play-core"` ; Basic Usage ■ The simplest version of the slider displays a single draggable handle to choose a numeric value between a default range of 0–100.

Ideal for scenarios like volume control, brightness adjustment, or progress selection. Size Variants

■ The slider supports multiple size options to fit different design requirements and layouts.

Smaller sliders suit compact UIs, while medium sizes provide comfortable interaction for most use cases. Available Size ■ `sm` (Small) : Compact slider ideal for dense layouts and mobile interfaces

`md` (Medium) : Standard size slider for most common use cases (default) States ■ Demonstrates the slider's different states, including disabled, active, and focused.

These states help communicate interactivity and status changes to users clearly. Multi Range

Slider ■ The multi-range version allows selection of both minimum and maximum values, offering a more flexible range selection experience.

It's ideal for use in filters, price sliders, or any range-based data input scenarios. Multi Range

Features ■ **Dual Handles** : Independent control of minimum and maximum values **Range**

Selection : Visual indication of selected range between handles **Collision Prevention** : Handles

cannot cross over each other **Synchronized Tooltips** : Both handles show their respective values

Input Type Variant ■ This variant combines a slider handle with a numeric input field, enabling precise manual entry alongside drag interaction.

It's especially useful in cases where exact numeric control is needed, such as filtering or budget

ranges. Input Type Features ■ **Dual Input Methods** : Users can drag the slider or type directly in

the input field **Real-time Sync** : Input field and slider stay synchronized **Validation** : Input respects

min/max boundaries and step values **Accessibility** : Input field provides keyboard navigation

alternative **Responsive Design** : Input field adapts to slider size variants **Icon Slider Variants** ■

Customizable slider with icon-based thumbs for enhanced visual feedback and thematic

consistency. **Icon Thumb Features** ■ **Custom Icons** : Replace default handle with Lucide icons

Thematic Consistency : Icons that match your content context **Multiple Variants** : Various icon

styles for different use cases **Responsive Sizing** : Icons scale appropriately with slider size **Color**

Theming : Icons inherit slider theme colors **Hover Effects** : Enhanced visual feedback on

interaction **Icon Thumb Variants** ■ **Volume Control** : Speaker/volume icons for audio controls

Brightness : Sun/brightness icons for display settings **Temperature** : Thermometer icons for

climate controls **Speed** : Gauge/speedometer icons for rate adjustments **Rating** : Star icons for

rating and review systems **Progress** : Arrow or progress icons for completion tracking **Orientation**

■ **Accessibility** ■ Built-in accessibility features ensuring WCAG compliance and inclusive user experience.

Accessibility Features

- **Keyboard Navigation** : Arrow keys, Home, and End key support
- **ARIA Attributes** : Proper role="slider" , aria-valuemin , aria-valuemax , aria-valuenow
- **Focus Management** : Clear focus indicators and outline
- **Touch Support** : Optimized for touch devices
- **Input Integration** : Numeric input field provides alternative input method

Keyboard Controls

- **Arrow Right/Up** : Increase value by step amount
- **Arrow Left/Down** : Decrease value by step amount
- **Home** : Jump to minimum value
- **End** : Jump to maximum value

API Reference

■ **Inputs**

Property	Type	Default	Description
min	number	0	Minimum value of the slider range
max	number	100	Maximum value of the slider range
value	number	0	Current value of the slider
step	number	1	Step increment for value changes
showTooltip	boolean	true	Whether to display the value tooltip
size	'sm' 'md' 'lg'	'md'	Size variant of the slider
type	'default' 'input'	'default'	Display type with or without input field
multiRange	boolean	false	Enable multi-range (two-handle) slider
minValue	number	20	Minimum selected value in multi-range mode
maxValue	number	80	Maximum selected value in multi-range mode
iconStart	string	"	Icon displayed at the start of the slider track
iconEnd	string	"	Icon displayed at the end of the slider track
handleIcon	string	"	Icon displayed on the slider handle
handleIconStart	string	"	Icon displayed on the start handle (multi-range)
handleIconEnd	string	"	Icon displayed on the end handle (multi-range)
customStyles	Record<string, string>	{}	CSS custom properties override
disabled	boolean	false	Disable the slider

Outputs

Event Type	Description
valueChange	EventEmitter<number> Emitted when the main slider value changes
minValueChange	EventEmitter<number> Emitted when the minimum value changes
maxValueChange	EventEmitter<number> Emitted when the maximum value changes

Methods

■ The component implements `ControlValueAccessor` for form integration:

Method	Parameters	Description
<code>writeValue</code>	value: number	Set value programmatically
<code>registerOnChange</code>	fn: Function	Register change callback
<code>registerOnTouched</code>	fn: Function	Register touched callback

CSS Custom Properties

■ The slider supports a wide range of CSS custom properties for theming and customization:

Property	Description
<code>--slider-container-height</code>	Height of the overall slider container
<code>--slider-container-gap</code>	Spacing between slider elements
<code>--slider-input-gap</code>	Gap between the slider track and input field
<code>--slider-size-sm-track-height</code>	Track height for small slider size variant
<code>--slider-size-sm-thumb-size</code>	Thumb size for small slider
<code>--slider-label-font-size-sm</code>	Label font size for small slider
<code>--slider-label-weight-sm</code>	Label font weight for small slider
<code>--slider-size-md-track-height</code>	Track height for medium slider size variant
<code>--slider-size-md-thumb-size</code>	Thumb size for medium slider
<code>--slider-label-font-size-md</code>	Label font size for medium slider
<code>--slider-label-weight-md</code>	Label font weight for medium slider
<code>--slider-track-height</code>	Height of the slider track
<code>--slider-track-background</code>	Background color of the slider track
<code>--slider-track-border-radius</code>	Border radius of the track
<code>--slider-progress-background</code>	Background color of the filled progress area
<code>--slider-progress-border-radius</code>	Border radius of the progress area
<code>--slider-thumb-size</code>	Size of the slider thumb
<code>--slider-thumb-border-radius</code>	Border radius of the thumb
<code>--slider-thumb-inner-background</code>	Background color inside the thumb
<code>--slider-thumb-shadow</code>	Shadow of the thumb
<code>--slider-thumb-shadow-hover</code>	Thumb shadow on hover
<code>--slider-focus-ring</code>	Style of the focus ring
<code>--slider-focus-ring-offset</code>	Offset distance of the

focus ring --slider-cursor Cursor style when hovering over slider --slider-tooltip-margin Margin around the tooltip --slider-tooltip-padding Padding inside the tooltip --slider-tooltip-border-radius Border radius of the tooltip --slider-value-color Text color of the tooltip value

--slider-label-font-family Font family used for labels --slider-label-line-height Line height for labels

--slider-mark-background Background color of slider marks --slider-handle-icon-width Width of the handle icon --slider-handle-icon-height Height of the handle icon --slider-input-width Width of the input field --slider-input-height Height of the input field --slider-input-padding Padding inside the input field --slider-input-border-radius Border radius of the input field --slider-input-border Border style of the input field --slider-input-background Background color of the input field

--slider-input-font-size Font size of the input text --slider-input-font-weight Font weight of the input text --slider-input-font-family Font family of the input text --slider-input-color Text color of the input

--slider-input-transition Transition style for input state changes --slider-input-focus-border-color Border color of input when focused --slider-input-hover-border-color Border color of input when hovered --slider-input-disabled-background Background color of disabled input

--slider-input-disabled-border-color Border color of disabled input --slider-value-color-disabled Text color for disabled value display --slider-disabled-color Color used in disabled state

--slider-disabled-rail-background Background of the slider rail when disabled

Best Practices

Implementation Guidelines

- Use appropriate step values for your use case (1 for integers, 0.1 for decimals)
- Set meaningful min/max boundaries that make sense for your data
- Consider hiding the tooltip for inline sliders in dense layouts
- Always provide proper labels for accessibility
- Choose appropriate size variants based on your layout density
- Use input type for scenarios requiring precise numeric input

Size Selection Guidelines

- Small : Use in compact layouts, mobile interfaces, or when space is limited
- Medium : Default choice for most applications and standard layouts

Input Type Usage

- Default Type : Best for visual-only interactions and quick value selection
- Input Type : Ideal for applications requiring precise numeric input or accessibility compliance

Form Integration

- Use reactive forms for complex validation scenarios
- Implement proper error handling and validation messages
- Consider debouncing frequent value changes for performance
- Leverage input type for better form accessibility and user experience

```
<aava-slider
  [value]="50"
  [min]="0"
  [max]="100"
  [step]="1"
  (valueChange)="onSliderChange($event)"
>
</aava-slider>
```

```
onSliderChange(value: number) {
  console.log('Single slider value:', value);
}
```

```
<aava-slider size="sm" [value]="30" [min]="0" [max]="100"> </aava-slider>
<aava-slider size="md" [value]="70" [min]="0" [max]="100"> </aava-slider>
```

```
<aava-slider [value]="50" [min]="0" [max]="100"> </aava-slider>
<h1>Normal State</h1>
```

```
<aava-slider [value]="50" [min]="0" [max]="100" [disabled]="true">
</aava-slider>
<h1>Disabled State</h1>
```

```
<aava-slider type="input" [value]="75" [min]="0" [max]="100"> </aava-slider>
<h1>Input Variant</h1>
```

```
<aava-slider [value]="50" [min]="0" [max]="100" [showTooltip]="false">
</aava-slider>
<h1>Without Tooltip</h1>
```

```

<aava-slider
  [multiRange]="true"
  [min]="0"
  [max]="100"
  [minValue]="minValue"
  [maxValue]="maxValue"
  (minValueChange)="onMinChange($event)"
  (maxValueChange)="onMaxChange($event)"
>
</aava-slider>

```

```

minValue = 20;
maxValue = 80;

onMinChange(value: number) {
  this.minValue = value;
  console.log('Min value changed:', value);
}

onMaxChange(value: number) {
  this.maxValue = value;
  console.log('Max value changed:', value);
}

```

```

<aava-slider
  type="default"
  [value]="50"
  [min]="0"
  [max]="100"
  [showTooltip]="true"
>
</aava-slider>
<h1>Default Type</h1>

```

```

<aava-slider type="input" [value]="75" [min]="0" [max]="100"> </aava-slider>
<h1>Input Type</h1>

```

```

currentValue = 50;

onSliderChange(value: number) {
  this.currentValue = value;
  console.log('Slider value:', value);
}

```

■ No code found

■ No code found